МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра информационных систем

Курсовая работа по дисциплине «Технологии программирования»

Разработка веб-приложения «Сервис поиска отелей Bochka»

09.03.02 Информационные системы и технологии

6 семестр 2023/2024 учебного года

Зав. кафедрой _____ к. т. н., доцент Д. Н. Борисов

Обучающийся _____ ст. 3 курса оч. отд. Порядин А.В.

Обучающийся _____ ст. 3 курса оч. отд. Исаченко Б.В.

Обучающийся _____ ст. 3 курса оч. отд. Ткаченко А.Ю.

Руководитель _____ ст. преподаватель Тарасов В.С.

Воронеж 2024

СОДЕРЖАНИЕ

Введение

В нашем современном мире, где путешествия стали неотъемлемой частью жизни многих людей, поиск подходящего места для проживания во время поездок становится всё более актуальной задачей. Сервисы бронирования отелей и других вариантов размещения играют ключевую роль в планировании путешествий и помогают экономить время и средства. Однако, в условиях растущей конкуренции и разнообразия предложений, выбор подходящего сервиса может стать настоящим испытанием.

Сервис поиска отелей Bochka направлен на решение этой проблемы, предоставляя пользователям удобный и функциональный инструмент для поиска и бронирования отелей по всему миру. Bochka отличается от других сервисов тем, что сочетает в себе широкий выбор предложений, конкурентоспособные цены и простоту в использовании.

Актуальность обусловлена необходимостью создания эффективного и удобного сервиса для поиска отелей, который помог бы путешественникам экономить время и средства, а также сделал бы процесс планирования поездок более приятным и удобным. Разработка сервиса Bochka отвечает этой потребности, предоставляя пользователям широкий выбор вариантов размещения и удобный инструмент для их поиска и бронирования.

Целью курсовой работы является разработка функционального сервиса поиска отелей Bochka. В рамках работы будет разработан пользовательский интерфейс, обеспечивающий удобное взаимодействие с сервисом, а также реализован функционал для поиска и бронирования отелей. Сервис Bochka будет способствовать повышению эффективности процесса планирования путешествий и поможет путешественникам найти подходящий вариант размещения в соответствии с их потребностями и бюджетом.

1 Постановка задачи

1.1 Цели создания приложения

Целями создания приложения являются:

— Создание системы, которая позволит пользователям легко и удобно бронировать отели для размещения во время путешествий или поездок;

— Увеличение доходов заказчика за счет продажи гостиничных номеров через онлайн-платформу;

— Продажа гостиничных номеров для конкретной аудитории, предпочитающей определенные условия размещения, такие как питание, наличие специальных удобств и т.д., что позволяет удовлетворить разнообразные потребности клиентов;

— Обеспечение информативной карточки каждого отеля с подробным описанием и фотографиями, что помогает пользователям принимать решения.

1.2 Задачи приложения

Приложение позволяет решать следующие задачи:

— Искать отели с глубокой фильтрацией по времени заселения, местоположению, наличию техники, парковке и другим параметрам;

— Просматривать подробную информацию о каждом отеле, включая фотографии, описание, услуги и прочее;

— Бронировать отель непосредственно через веб-сайт;

— Сохранять историю бронирования для последующего доступа;

— Создавать учётную запись пользователя;

— Создавать учётную запись админа для добавления отелей в систему;

1.3 Требования к приложению

1.3.1 Требования к приложению в целом

Разрабатываемое приложение должно удовлетворять следующим основным требованиям:

— Приложение должно корректно работать в современных веб-браузерах;

— Приложение должно реализовывать основные функциональные задачи, соответствующие целям проекта;

— Созданное приложение должно иметь архитектуру, соответствующую шаблону клиент-серверного приложения, с разделением на back-end и front-end;

— Взаимодействие между back-end и front-end должно осуществляться посредством REST API.

1.3.2 Требования к функциям (задачам), выполняемым приложением

Разрабатываемое приложение должно соответствовать следующим функциональным требованиям:

Неавторизованный пользователь должен обладать возможностью:

— Авторизоваться/зарегистрироваться в приложении;

— Получать информацию о предложениях отелей с глубокой фильтрацией по различным критериям, таким как местоположение, цена, удобства и другие параметры;

— Просматривать детальную информацию о каждом отеле, включая фотографии и описания;

— Выполнять поиск отелей по различным критериям.

Авторизованный пользователь (в роли клиента) должен обладать возможностью:

— Получать информацию о предложениях отелей с глубокой фильтрацией по различным критериям, таким как местоположение, цена, удобства и другие параметры;

— Просматривать детальную информацию о каждом отеле, включая фотографии и описания;

— Выполнять поиск отелей по различным критериям;

— Просматривать свою историю бронирования;

— Бронировать отель;

Авторизованный пользователь (в роли админа) должен обладать возможностью:

— Добавлять информацию об отеле, включая описание, фотографии, цены и доступные удобства;

— Обновлять информацию об отеле, в том числе актуализация цен и доступности номеров;

— Просматривать и управлять бронированиями;

1.3.3 Требования к оформлению и верстке страниц

Все страницы сайта должны быть выполнены в едином стиле, соответствующем тематике отельного бронирования. Цветовая палитра и стили шрифтов должны быть гармонично подобраны и привлекательны для пользователей. Приложение должно содержать разработанный логотип, отражающий его назначение и стиль.

Необходимо корректное и одинаковое отображение страниц сайта в следующих браузерах:

— Google Chrome 122.0.6261.128/129;

— Yandex Browser 23.11.3.955;

— Microsoft Edge 121.0.2277.83;

— Safari 16.5.2;

— Mozilla Firefox 123.0.1.

Верстка сайта должна быть адаптирована под популярные разрешения экранов, чтобы обеспечить удобство использования и приятный внешний вид для всех пользователей.

1.3.4 Требования к защите информации

Для обеспечения безопасности информации будет использоваться механизм JWT-токенов. Даже в случае получения злоумышленником такого токена, который предоставляет доступ ко всем функциям приложения, его действие будет ограничено заданным периодом времени, после чего токен станет недействительным и потребуется получить новый.

1.4 Задачи, решаемые в процессе разработки

Были поставлены следующие задачи:

— Анализ предметной области;

— Обзор аналогов;

— Постановка задачи;

— Создание репозитория GitHub и доски в GitHub Projects;

— Разработка требований: к приложению в общем, к функциям, к структуре, к программному обеспечению, к оформлению и верстке страниц, к защите информации;

— Создание диаграмм: прецедентов, состояний, активностей, последовательностей, классов, развертывания;

— Разработка дизайна приложения;

— Написание технического задания в соответствии с ГОСТ 34.602 – 2020;

— Реализация интерфейса приложения;

— Реализация серверной части приложения;

— Развертывание приложения;

— Написание курсовой работы.

2 Анализ предметной области

2.1 Глоссарий

В настоящей работе используются следующие термины и сокращения с соответствующими определениями:

- **API** – программный интерфейс приложения. Описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой;
- **Frontend** – это клиентская часть продукта (интерфейс, с которым взаимодействует пользователь);
- **Backend** – программно-аппаратная часть приложения (логика приложения, скрытая от пользователя);
- **Авторизация** – Предоставление определённому лицу прав на выполнение определённых действий; а также процесс проверки (подтверждения) данных прав при попытке выполнения этих действий;
- **Авторизованный пользователь** – Пользователь, который успешно прошел процесс авторизации в приложении, предоставив свои учетные данные и подтвердив свою идентичность. Авторизованный пользователь имеет доступ ко всем основным функциям приложения;
- **Глубокая фильтрация** – функциональность в веб-приложениях, которая позволяет пользователям настраивать широкий спектр параметров для поиска и отображения результатов, соответствующих их конкретным потребностям и предпочтениям;
- **Искусственный интеллект** – Набор технологий и алгоритмов, которые позволяют приложению анализировать большие объемы данных, выявлять закономерности и предсказывать поведение

пользователей для персонализации рекомендаций и улучшения качества обслуживания;

— **Неавторизованный пользователь** – Пользователь, который еще не прошел процесс авторизации в приложении или не предоставил верные учетные данные для подтверждения своей идентичности. Неавторизованный пользователь имеет ограниченный доступ к функциям приложения;

— **Профиль (в веб-приложении)** – Учетная запись пользователя в веб-приложении, вход в которую осуществляется с помощью логина / номера телефона / e-mail и пароля. В учетной записи содержится информация о пользователе;

— **Серверная** часть – это программа, которая обеспечивает взаимодействие клиента и сервера;

— **Сервер** – это устройство, в частности компьютер, которое отвечает за предоставление услуг, программ и данных другим клиентам посредством использования сети;

— **Фреймворк** – Программные продукты, которые упрощают создание и поддержку технически сложных или нагруженных проектов. Фреймворк, как правило, содержит только базовые программные модули.

2.2 Обзор аналогов

2.2.1 Ostrovok

"Ostrovok" является одним из крупнейших российских онлайн-сервисов для бронирования гостиниц и других вариантов размещения. Сервис предлагает широкий выбор отелей и апартаментов как в России, так и за рубежом.

Достоинства:

— Обширная база отелей и различных типов жилья во множестве городов и стран;

— Подробные описания и отзывы о каждом объекте размещения от реальных гостей, что помогает пользователям принимать информированные решения;

— Гибкая система фильтрации и поиска, позволяющая настраивать запросы в соответствии с предпочтениями пользователей;

— Частые акции и скидки, которые могут существенно снизить стоимость проживания.

Недостатки:

— Некоторым пользователям может показаться, что процесс бронирования требует слишком много шагов;

— Возможность встретить неактуальную или недостоверную информацию о жилье или рейсах, так как отзывы и данные могут устаревать;

— Недостаток персонализации, поскольку сервис иногда предлагает одни и те же варианты для всех пользователей, не учитывая их индивидуальных предпочтений.

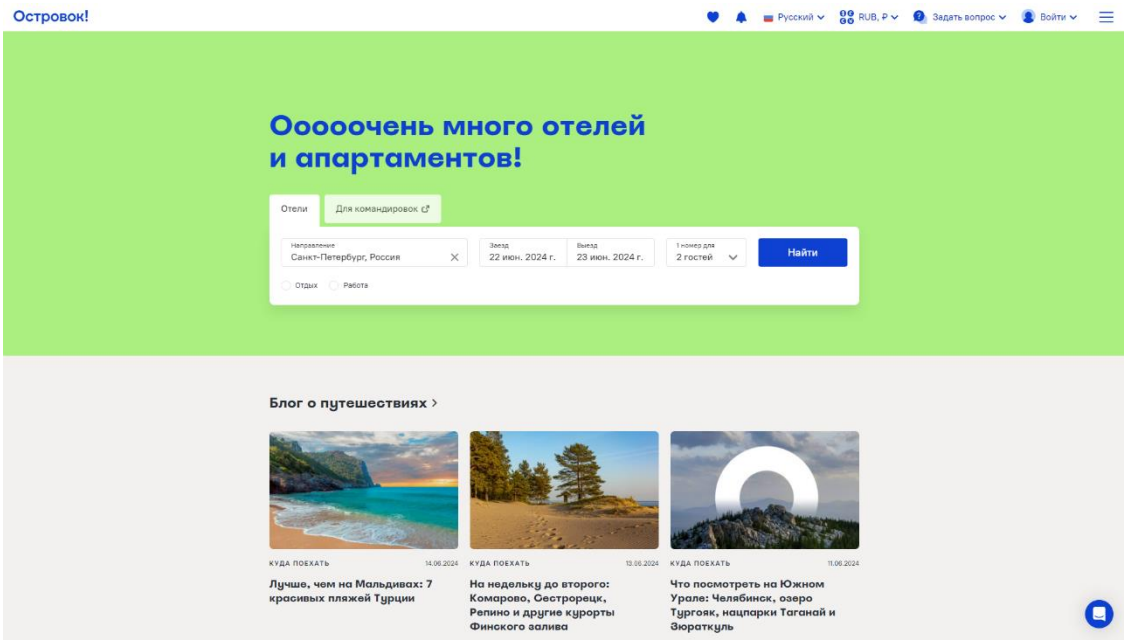

Рисунок 1 - Главная страница сервиса Ostrovok

2.2.2 OneTwoTrip

"OneTwoTrip" — это онлайн-сервис, предоставляющий услуги бронирования отелей, авиабилетов и других туристических услуг. Сервис ориентирован на широкий круг путешественников и предлагает комплексные решения для планирования поездок.

Достоинства:

— Широкий выбор отелей, авиабилетов и пакетных туров, что позволяет планировать поездку в одном месте;

— Возможность накапливать бонусы и использовать их для оплаты следующих бронирований;

— Интерактивные фильтры и карты для поиска жилья и рейсов в определенных районах или по близости к конкретным объектам.

Недостатки:

— Некоторым пользователям может быть неудобно пользоваться сервисом из-за большого количества рекламных предложений и уведомлений;

— Меньше вариантов для бронирования жилья и билетов за границей по сравнению с международными сервисами.

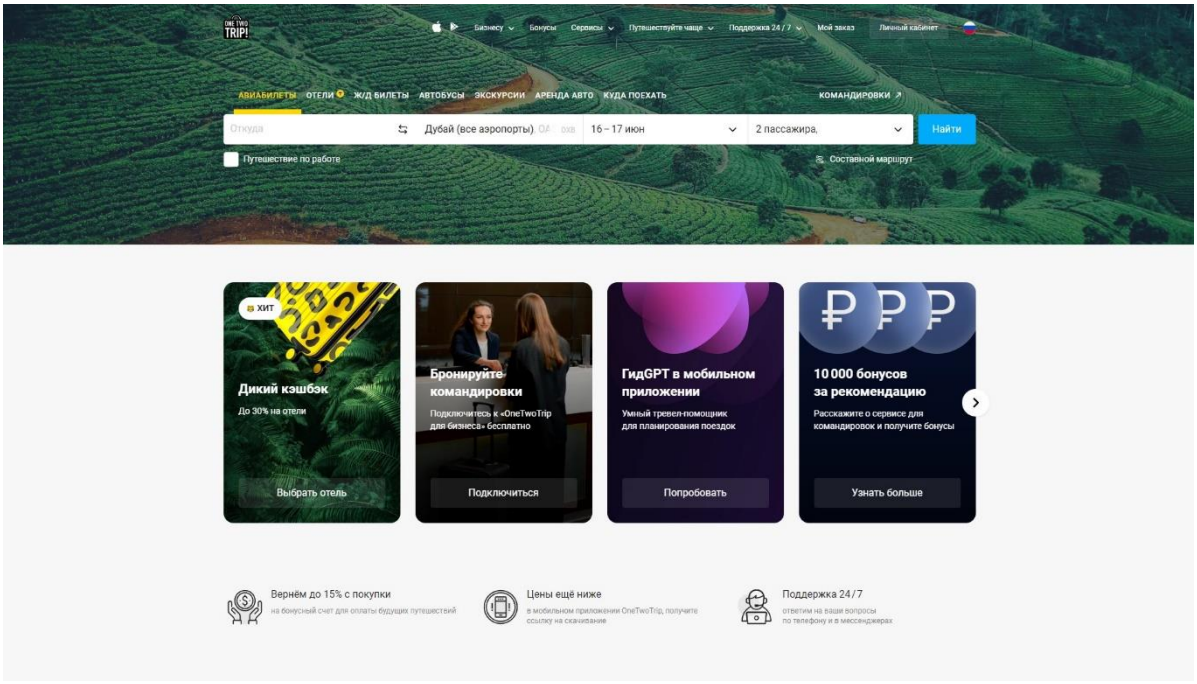

Рисунок 2 - Главная страница сервиса OneTwoTrip

2.2.3 Яндекс.Путешествия

"Яндекс.Путешествия" представляет собой сервис для планирования и организации путешествий, который объединяет информацию о бронировании отелей, покупке авиабилетов, аренде автомобилей и других важных аспектах путешествия.

Достоинства:

— Интеграция с другими сервисами "Яндекса", такими как "Яндекс.Карты" и "Яндекс.Маршруты", что обеспечивает удобное планирование маршрутов и перемещений.

— Широкий выбор предложений от различных партнеров, включая отели, авиакомпании, агентства аренды автомобилей и турагентства;

— Возможность накапливать бонусы и кэшбэк, которые можно использовать для оплаты следующих бронирований.

Недостатки:

— Некоторым пользователям может показаться, что интерфейс перегружен информацией из-за большого количества предложений и уведомлений;

— Недостаточная персонализация и индивидуализация рекомендаций, что может привести к неоптимальному выбору для некоторых пользователей;

— Ограниченный выбор международных направлений по сравнению с глобальными сервисами.

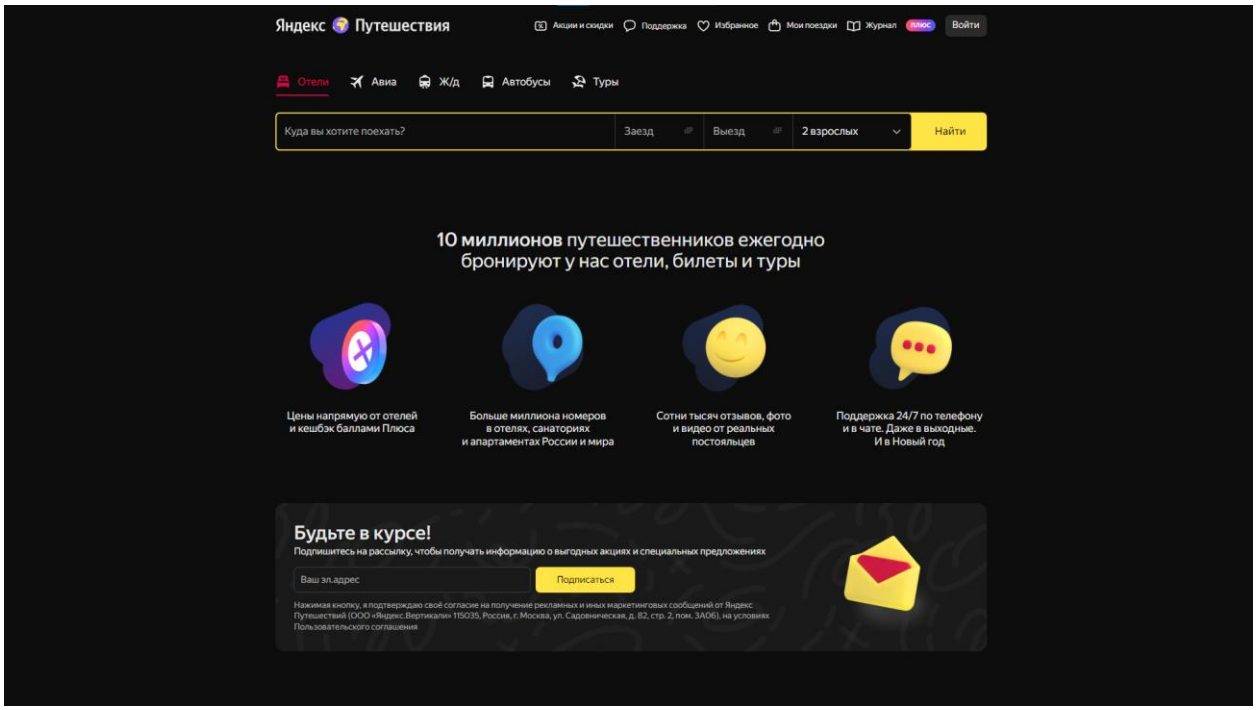

Рисунок 3 - Главная страница сервиса Яндекс.Путешествия

3 Моделирование системы

3.1 Диаграмма прецедентов

Рассмотрим полную диаграмму использования приложения различными типами пользователей (Рисунок 4). В данном контексте составление диаграммы прецедентов обусловлено необходимостью моделирования системы и понимания её функциональности и потребностей пользователей. Эти диаграммы помогают определить основные действия, которые пользователь должен совершить в системе для достижения определенных целей. Они также помогают выявить возможные риски и проблемы, которые могут возникнуть в процессе использования системы.

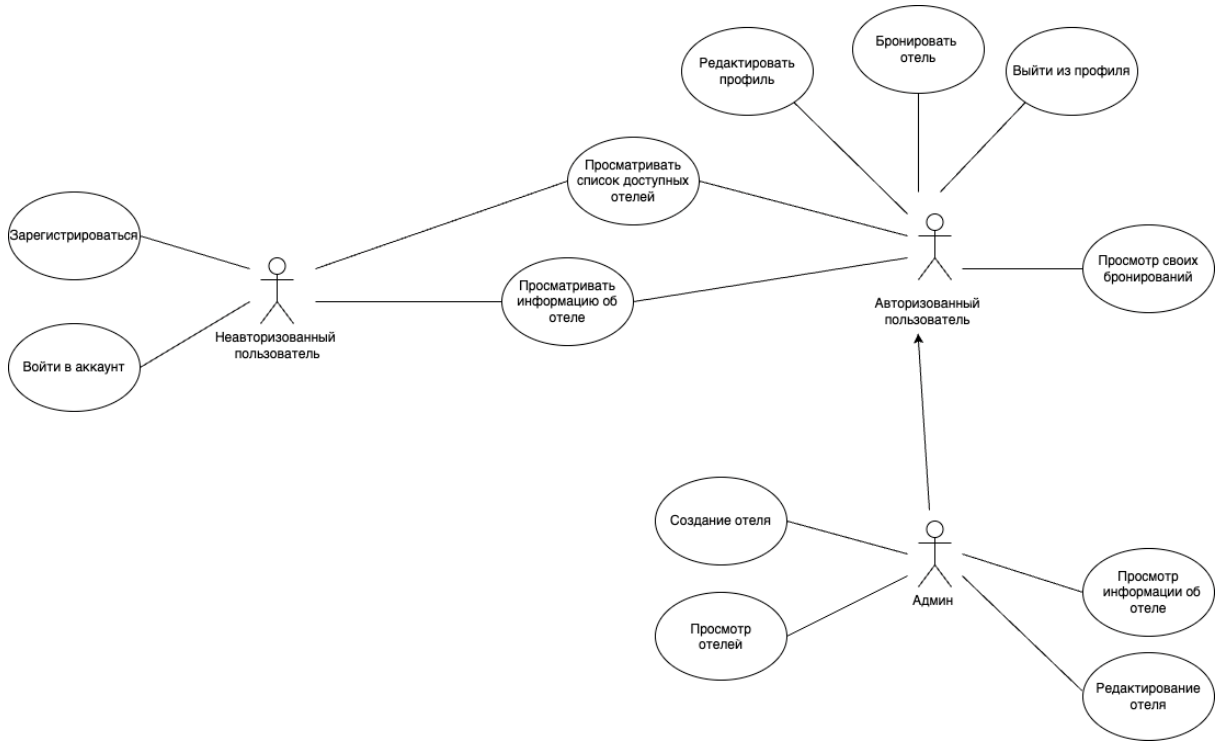

Рисунок 4 - Диаграмма прецедентов (use-case)

3.2 Диаграмма последовательности

Диаграмма последовательности (Рисунки 5 - 7) играет важную роль в проекте, помогая более глубоко понять процесс, повысить его эффективность и упростить взаимодействие.

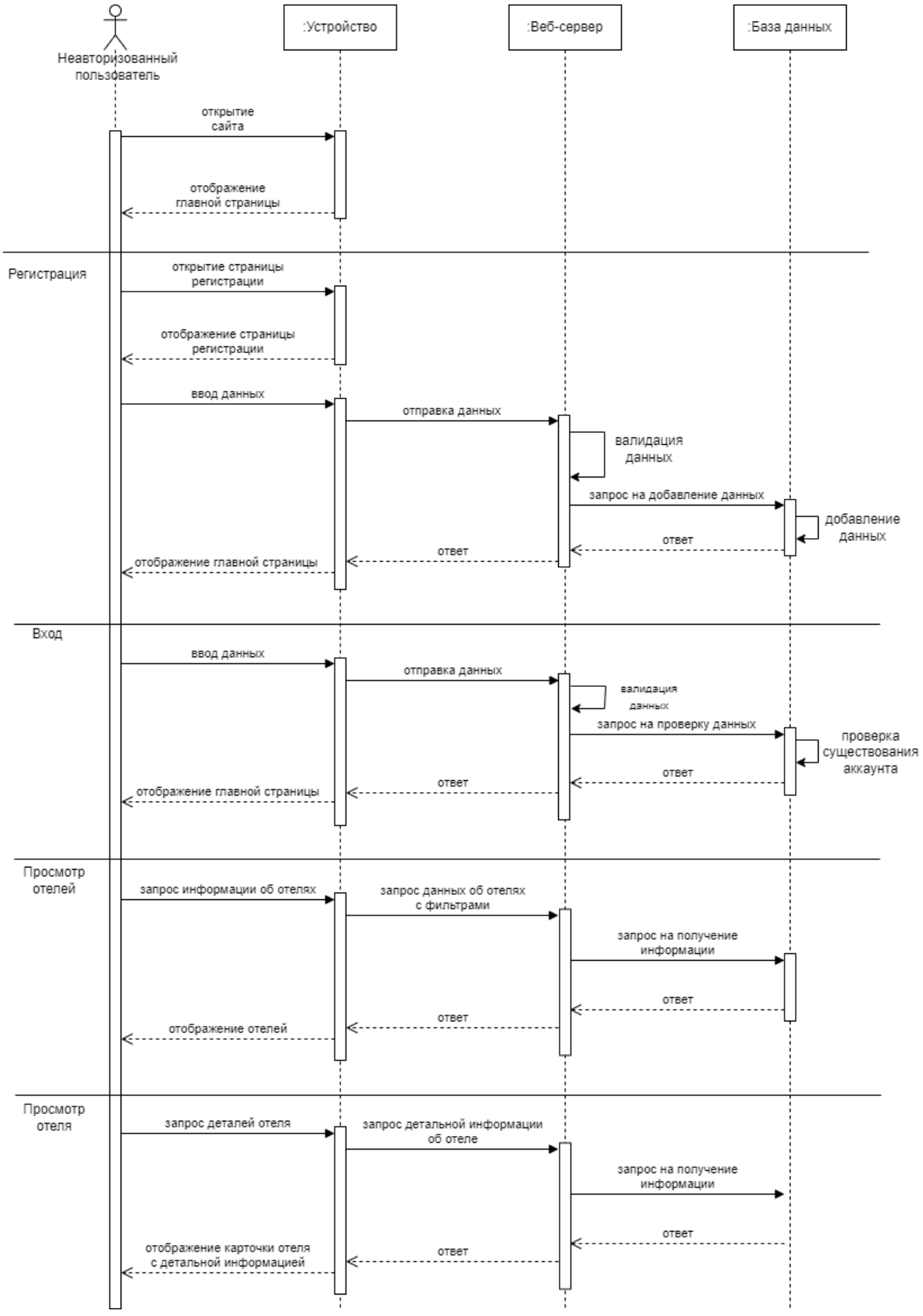

Рисунок 5 - Диаграмма последовательности неавторизованного пользователя (sequence)

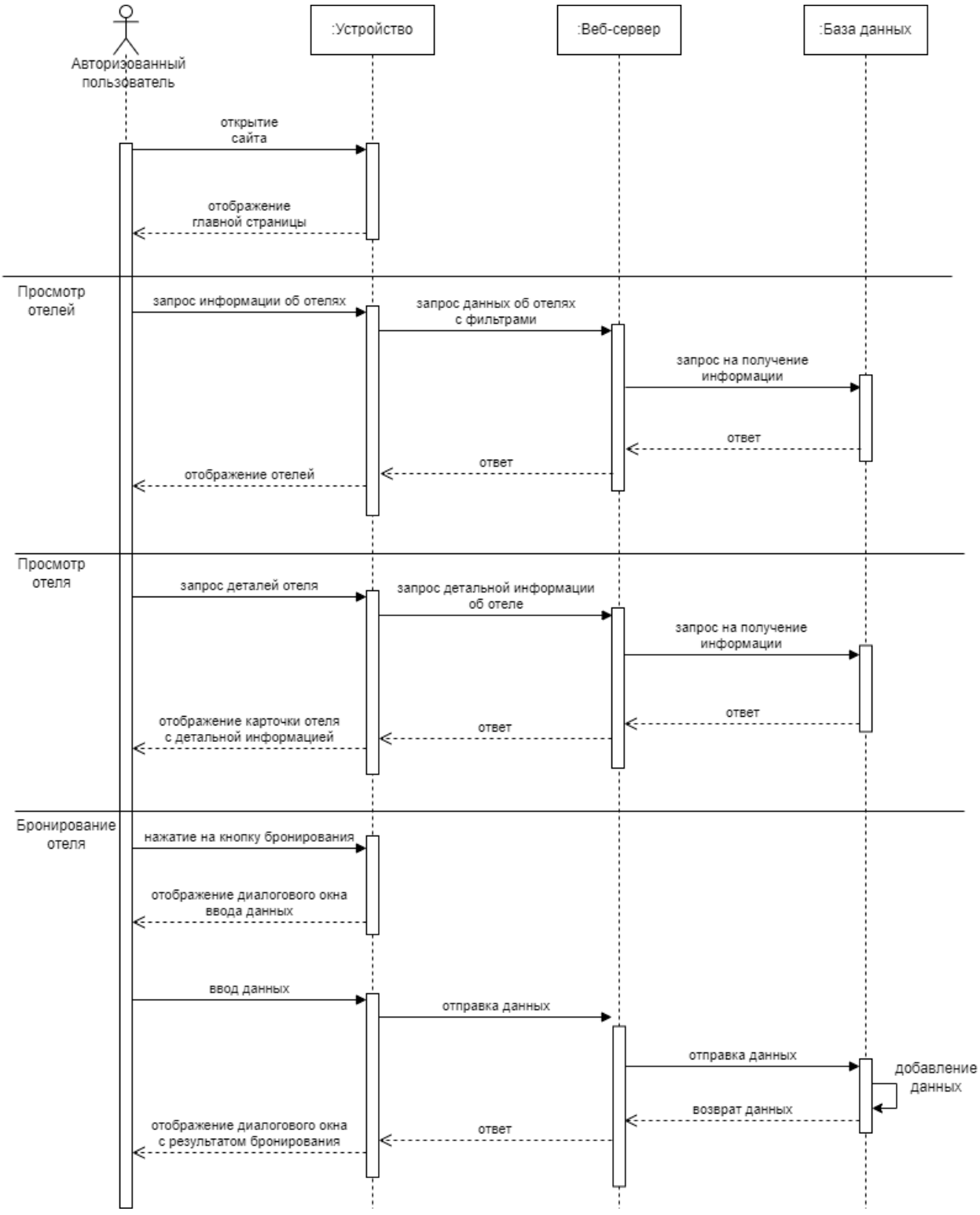

Рисунок 6 - Диаграмма последовательности клиента (sequence)

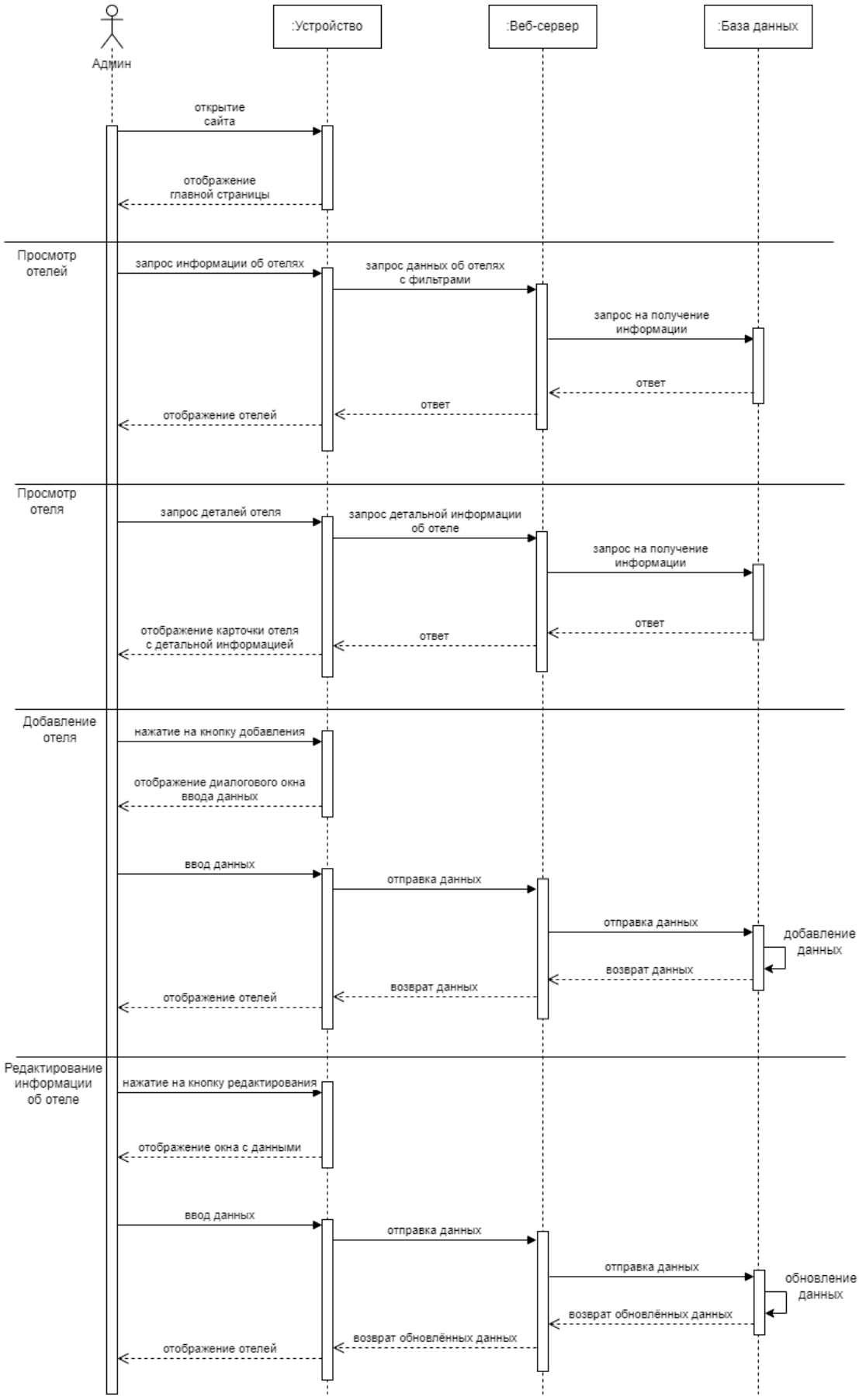

Рисунок 7 - Диаграмма последовательности админа (sequence)

3.3 Диаграмма развёртывания

Диаграмма развёртывания (Рисунок 8) помогает определить необходимости в аппаратном обеспечении, спланировать установку и настройку компонентов системы, а также оценить её производительность и масштабируемость.

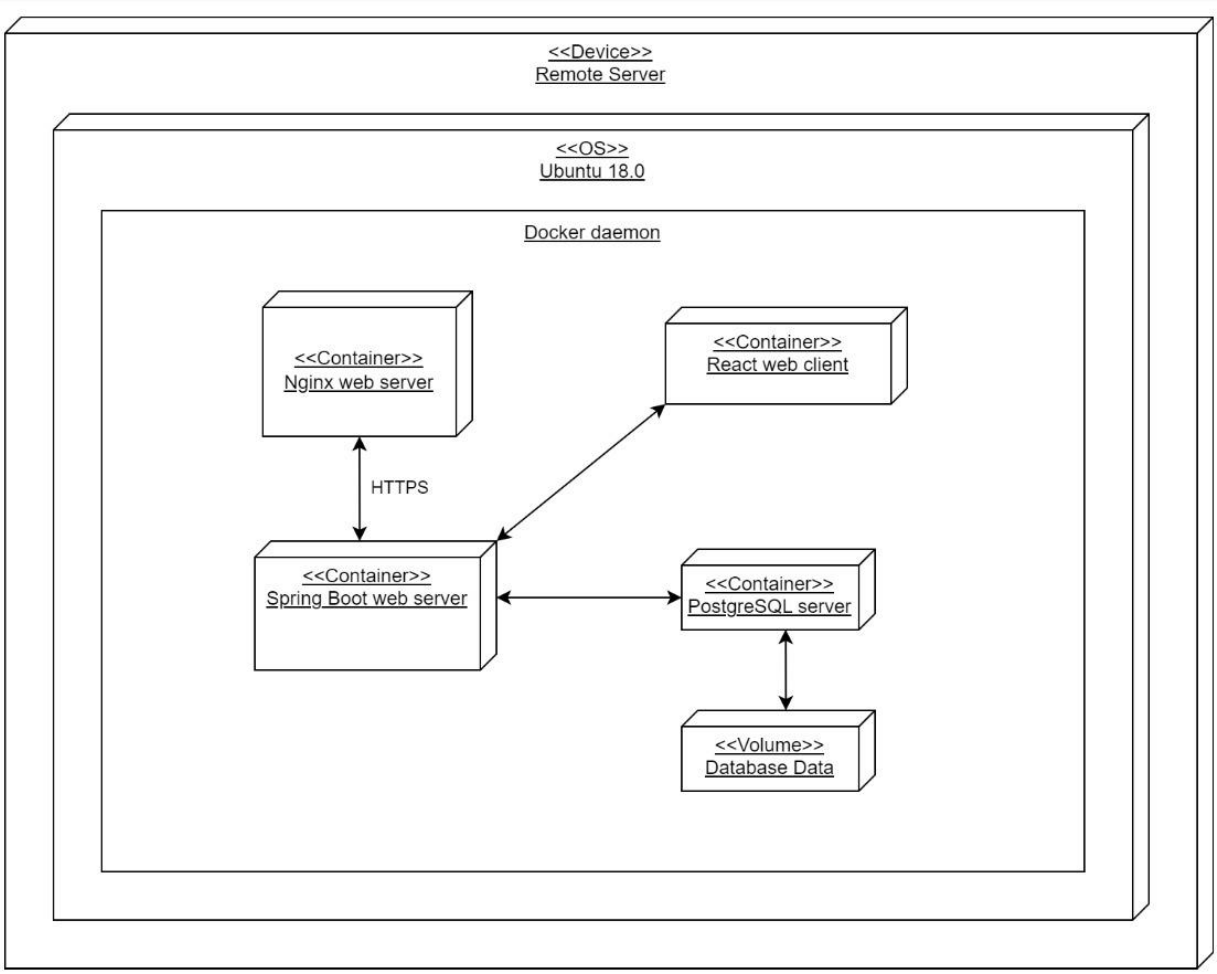

Рисунок 8 - Диаграмма развёртывания приложения (deployment)

3.4 Диаграммы состояния

Диаграмма состояния (Рисунки 9 - 11) позволяет анализировать возможные сценарии поведения системы, выделять ключевые состояния и переходы между ними, а также оценивать её надежность и устойчивость к ошибкам. В рамках нашего проекта были разработаны три диаграммы состояний для неавторизованного пользователя, клиента и админа.

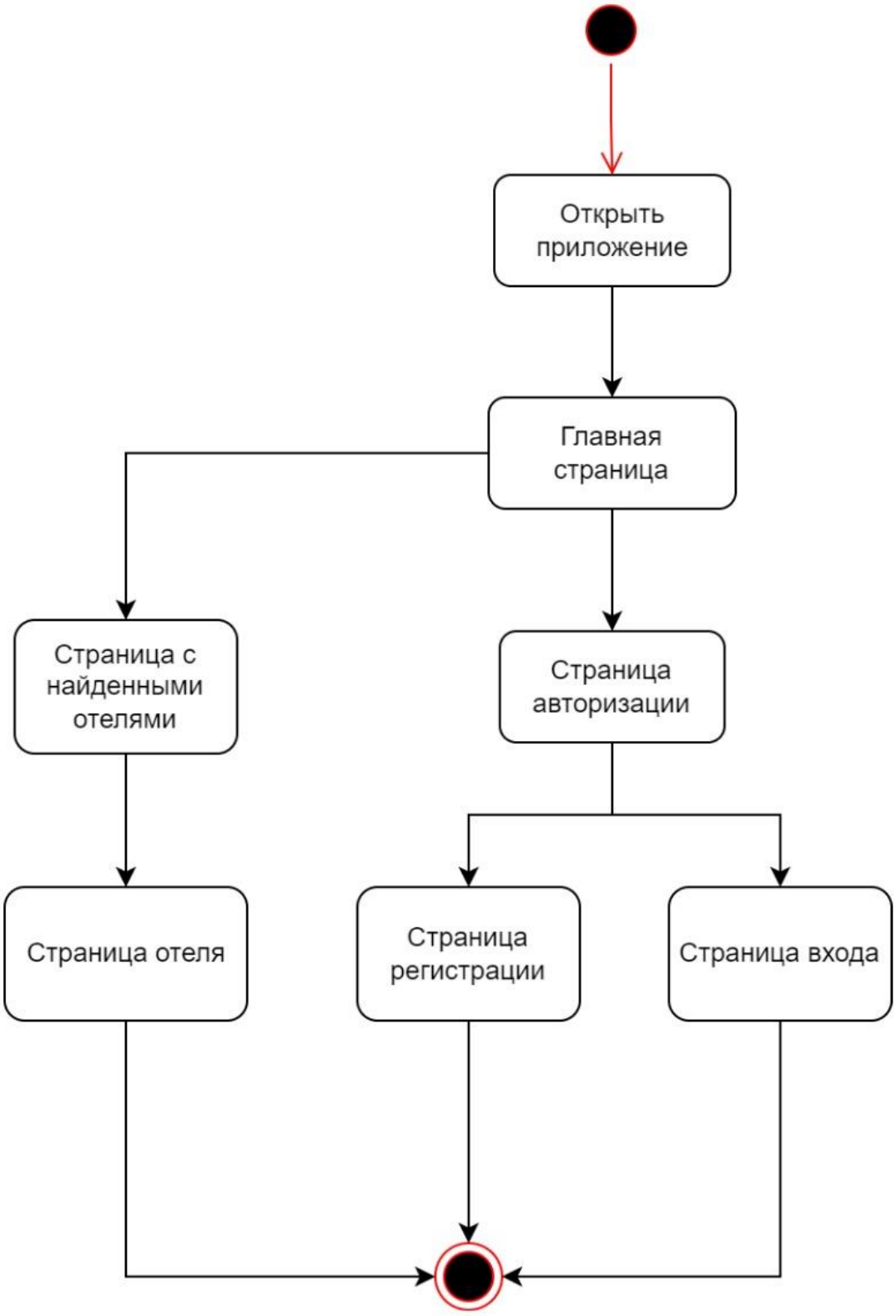

Рисунок 9 - Диаграмма состояния неавторизованного пользователя (statechart)

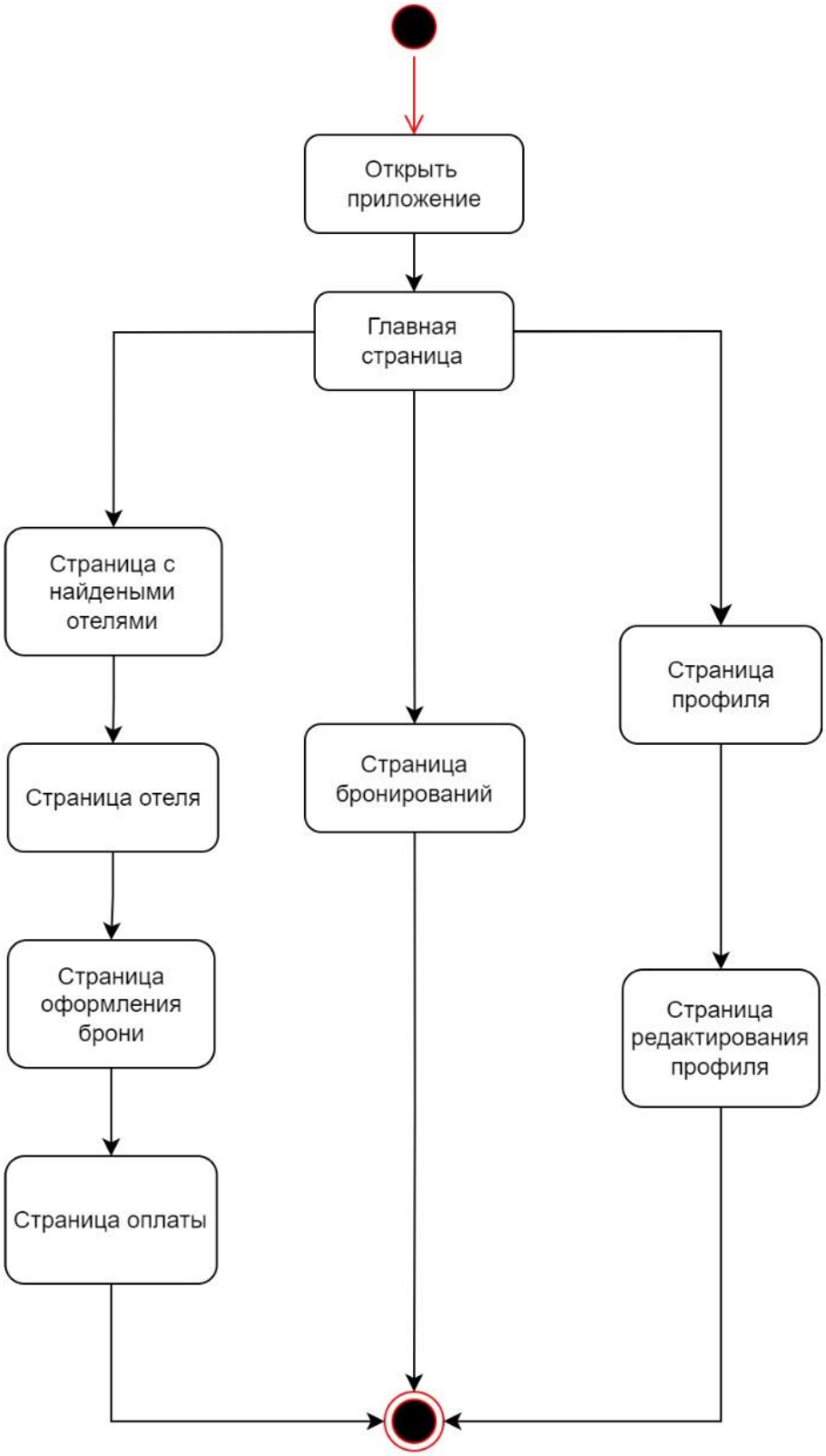

Рисунок 10 - Диаграмма состояния клиента (statechart)

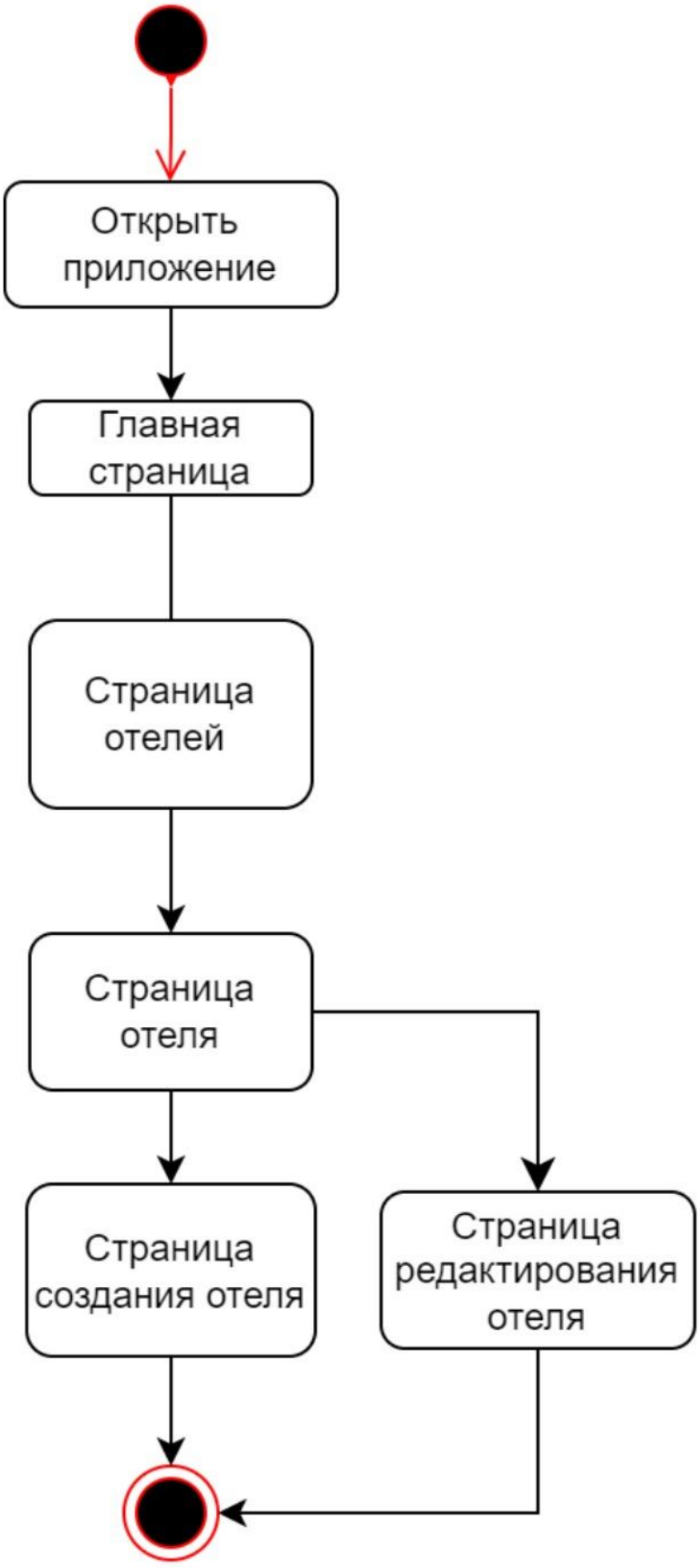

Рисунок 11 - Диаграмма состояния админа (statechart)

3.5 ER-диаграмма

ER-диаграмма (Рисунок 12) предоставляет структурное представление данных, иллюстрируя сущности (объекты) в системе и их взаимосвязи. Она помогает определить основные сущности, их атрибуты и отношения между ними, что облегчает проектирование базы данных и анализ требований к системе.

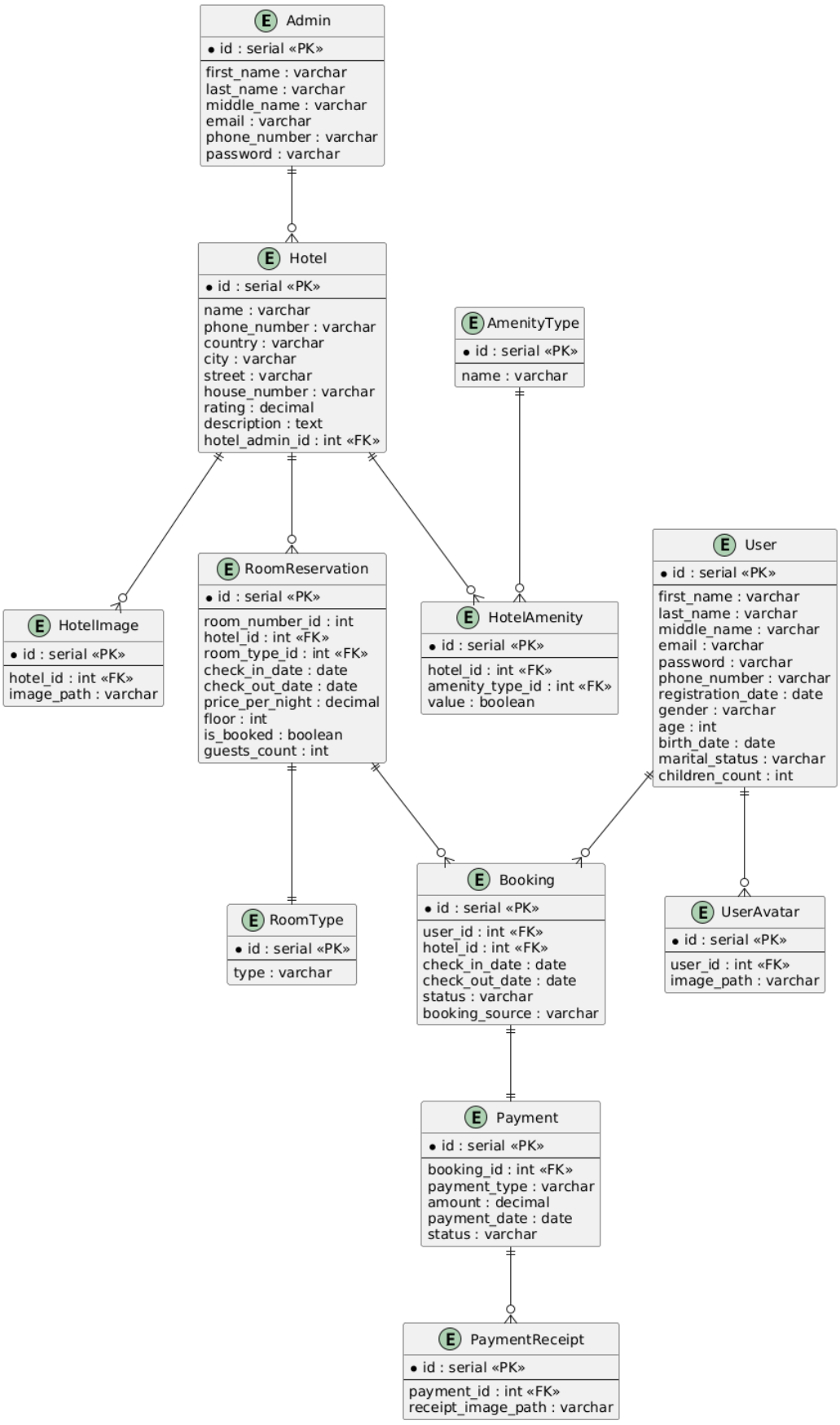

Рисунок 12 - ER-диаграмма

3.6 Диаграммы активности

Диаграмма активности (Рисунок 13) помогает улучшить понимание системных процессов, выявить и оптимизировать узкие места. Кроме того, она применяется для описания бизнес-процессов и управления проектами.

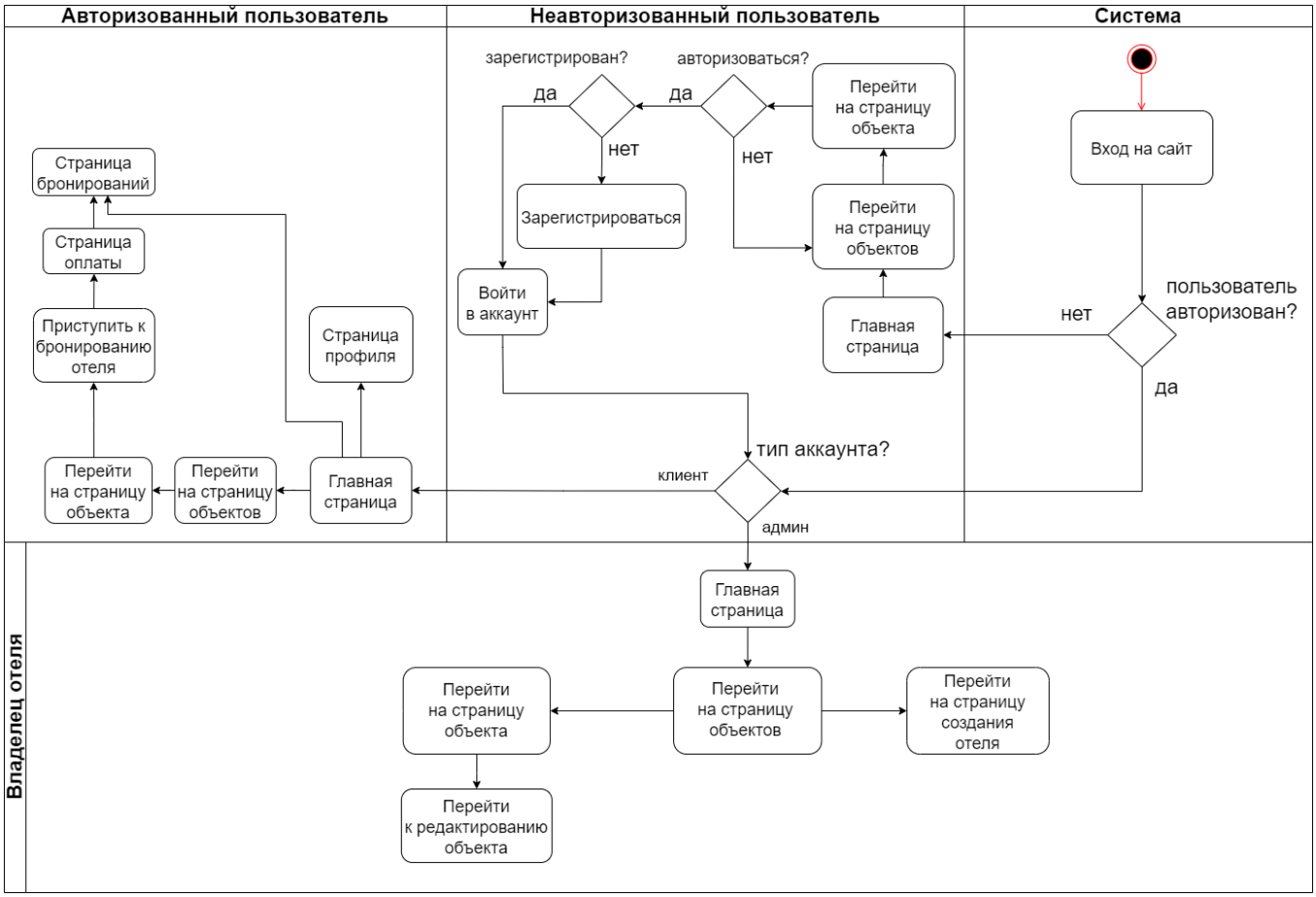

Рисунок 13 - Диаграмма активности

4 Реализация

4.1 Средства реализации

Созданное веб-приложение имеет архитектуру, соответствующую шаблону клиент-серверного приложения, с разделением на back-end и front-end. Обмен между ними осуществляется посредством REST API.

В процессе разработки был использован Git-flow для организации работы. В нашем проекте эта модель ветвления включает две основные ветки: main и develop, а также временные ветки для добавления новых функций.

4.1.1 Module

Клиентская часть разрабатывалась в соответствии с архитектурным паттерном Module — паттерн, который используется для организации кода в отдельные модули или компоненты. Цель использования паттерна Module - избежать конфликтов и обеспечить лучшую структурированность, масштабируемость и повторное использование кода. Каждый модуль содержит свою собственную область видимости, что позволяет исключить конфликты между переменными или функциями из разных модулей.

4.1.2 MVC

Серверная часть разрабатывалась в соответствии с подходом MVC (Model – View – Controller) — паттерн разработки, разделяющий архитектуру приложения на три модуля: модель (Model), представление или вид (View), контроллер (Controller).

— Model – это основная логика приложения. Отвечает за данные, методы работы с ними и структуру программы. Модель реагирует на команды из контроллера и выдает информацию и/или изменяет свое состояние. Она передает данные в представление;

— View – отвечает за визуализацию информации, которую он получает от модели. View отображает данные на уровне пользовательского интерфейса. Например, в виде таблицы или списка. Представление

определяет внешний вид приложения и способы взаимодействия с ним;

— Controller – обеспечивает взаимодействие с системой: обрабатывает действия пользователя, проверяет полученную информацию и передает ее модели. Контроллер определяет, как приложение будет реагировать на действия пользователя. Также контроллер может отвечать за фильтрацию данных и авторизацию.

4.1.3 Java

Для реализации серверной части приложения использовался язык программирования Java 17 и фреймворк Spring Boot 3.3.2

Язык программирования Java характеризуется строгой типизацией и богатым синтаксисом, что делает его эффективным инструментом для разработки масштабируемых приложений. Он предоставляет широкие возможности для создания многоуровневых приложений благодаря поддержке объектно-ориентированного программирования, а также предлагает обширную стандартную библиотеку для работы с данными, сетью и файлами.

Фреймворк Spring Boot предоставляет удобные инструменты для разработки веб-приложений и REST API. Он поддерживает микросервисную архитектуру, обеспечивает высокую производительность и ускоряет процесс разработки за счёт встроенной конфигурации и автоматизации многих процессов.

4.1.4 Swagger

Для документирования и тестирования API используется инструмент Swagger. Он позволяет автоматически генерировать и поддерживать актуальную документацию API на основе аннотаций в коде. Swagger предоставляет интерактивный интерфейс, который упрощает тестирование API прямо из браузера. Интеграция Swagger с Spring Boot обеспечивает

простоту использования и эффективность в разработке, позволяя разработчикам быстро просматривать и проверять функциональность API.

4.1.5 PostgreSQL

Для хранения данных используется реляционная база данных PostgreSQL 16.2. Благодаря своей высокой производительности, она обеспечивает возможность значительного расширения клиентской базы в долгосрочной перспективе без необходимости перехода на другие технологии.

4.1.6 JavaScript

Для реализации клиентской части приложения использовался язык программирования JavaScript и библиотека React 18.3.1

JavaScript — это многофункциональный язык программирования, который широко применяется для создания динамических и интерактивных веб-приложений. Он обладает гибким синтаксисом и широкой экосистемой инструментов, что делает его одним из наиболее популярных языков для веб-разработки.

React — это библиотека JavaScript для создания пользовательских интерфейсов. Она позволяет разбивать интерфейс на маленькие, переиспользуемые компоненты, что упрощает разработку и обновление веб-приложений.

4.1.7 Nginx

Для обеспечения поддержки SSL и проксирования запросов к back-end приложению использовался веб-сервер Nginx.

Nginx представляет собой мощный и гибкий веб-сервер, который обладает высокой производительностью и надежностью. Он предоставляет возможности для настройки виртуальных хостов, обработки статических и динамических контентов, а также обеспечивает эффективное проксирование запросов к back-end приложению.

4.1.8 Docker

Для развёртывания приложения использовался Docker версии 26.1.1.

Docker - это средство для контейнеризации приложений, что делает их переносимыми и ускоряет процесс разработки. За счет изоляции процессов, приложение не влияет на внешнее окружение, что повышает безопасность.

Инструмент Docker Compose позволяет легко настроить запуск нескольких контейнеров за счёт декларативного описания запускаемых сервисов и сетей.

4.1.9 AppMetrica

Для сбора информации о работе приложения, пользовательских действиях и аналитики применяется сервис AppMetrica. Благодаря удобному SDK и гибким настройкам, этот сервис позволяет легко настраивать отправку событий пользователей и создавать аналитические отчёты, такие как воронки конверсий, на основе собранных данных.

4.2 Разработка frontend

Клиентская часть приложения была написана на языке JavaScript с использованием библиотеки React. Для отправки запросов с клиентской части приложения на серверную часть используется билиотека axios. Она предоставляет интерфейс для выполнения HTTP-запросов прямо из браузера.

Для создания модульной и переиспользуемой архитектуры, приложение было разбито на компоненты. Каждый компонент отвечает за конкретную функциональность или отображение определенной части пользовательского интерфейса.

Проект имеет следующую структуру (Рисунок 14):

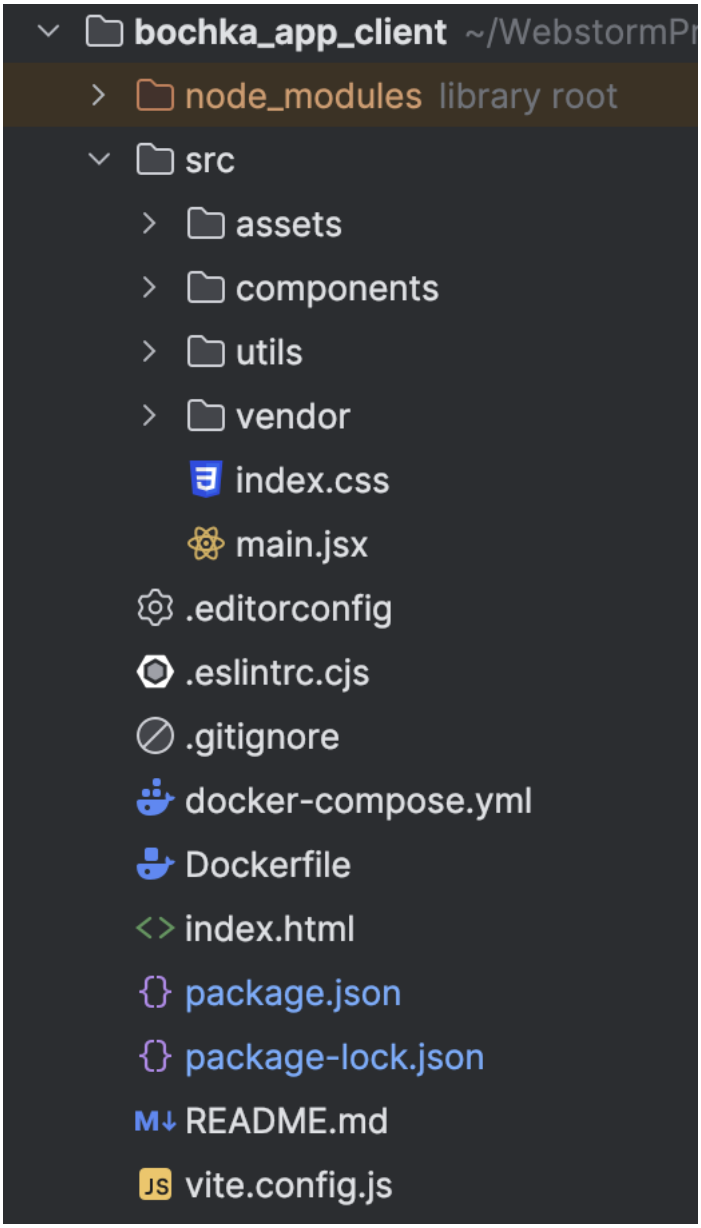

Рисунок 14 - Структура клиентской части приложения

— Модуль «assets». Этот модуль содержит все статические ресурсы, такие как изображения (img) и шрифты (fonts);

— Модуль «components». Этот модуль содержит компоненты, которые используются в приложении;

— Модуль «utils». Этот модуль содержит константы и класс для связи с api;

— Модуль «vendor». Этот модуль содержит стили для подключения шрифтов и приведения первоначальных стилей в единый вид на всех устройствах;

— Файл «main,jsx». Этот файл является точкой входа в приложение. Он служит главным файлом, с которого начинается выполнение приложения.

4.3 Разработка backend

Серверная часть приложения была написана на языке Java с использованием фреймворка Spring Boot.

Проект имеет следующую структуру и модули (Рисунок 15):

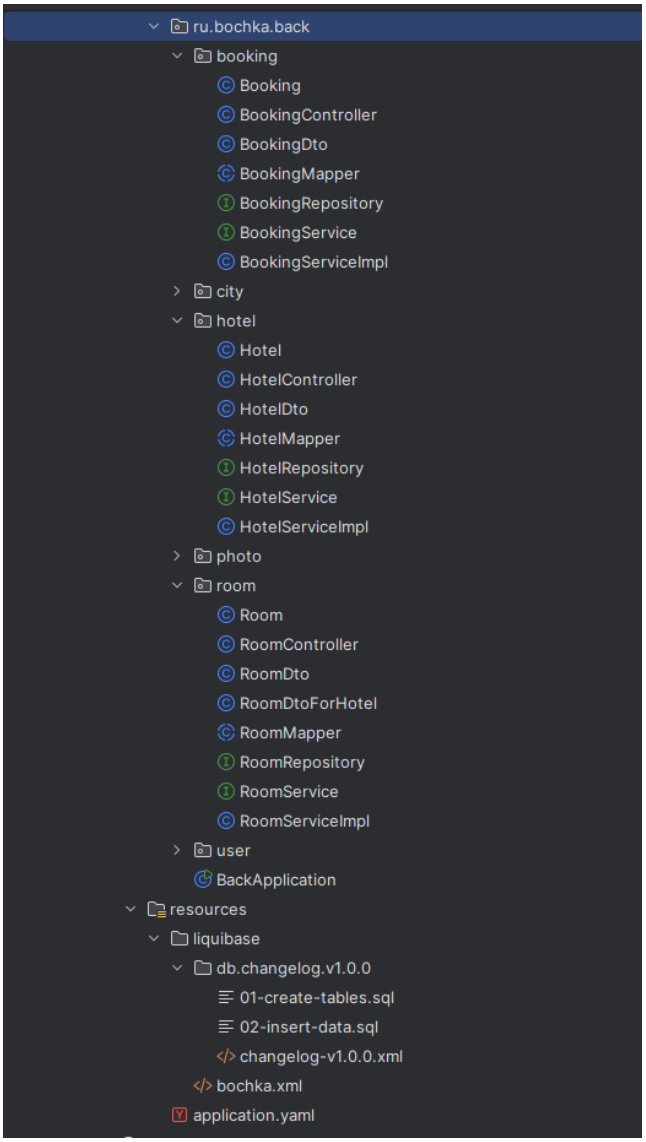

Рисунок 15 - Структура серверной части приложения

В каждом из пакетов «booking» , «city», «hotel», «photo», «room», «user» реализованы следующие основные компоненты:

— Контроллеры - классы, которые отвечают за обработку HTTP-запросов и описание эндпоинтов.

— Модели DTO - объекты передачи данных, которые используются для обмена информацией между слоями приложения.

— Репозитории – интерфейсы, которые предоставляют методы для выполнения основных операций.

— Мапперы – классы, преобразующие данные между различными слоями приложения.

— Сервисный слой – классы, в которых реализована бизнес-логика приложения.

Пакет «resources» содержит конфигурационные файлы, необходимые для работы приложения. Основные из них:

— «application.yaml», в котором указаны настройки базы данных, порты и другие переменные среды.

— Папка «liquibase», которая содержит SQL-скрипты миграции базы данных.

Класс «BackApplication.java» является точкой входа в приложение Spring Boot. Он запускает встроенный веб-сервер, инициирует создание всех необходимых компонентов и запускает приложение.

Для развёртывания приложения был использован виртуальный сервер (VPS) от провайдера TimeWeb. В качестве веб-сервера был выбран Nginx. Приложение было развёрнуто на этом сервере, а Nginx использован для обслуживания статического контента и проксирования запросов к приложению.

Для описания спецификации API использовался Swagger. Его интеграцию в проект обеспечивает Spring Boot.

4.4 Основная функциональность сайта

4.4.1 Поиск отелей

При входе на сайт пользователю открывается главная страница сайта со строкой поиска (Рисунок 16).

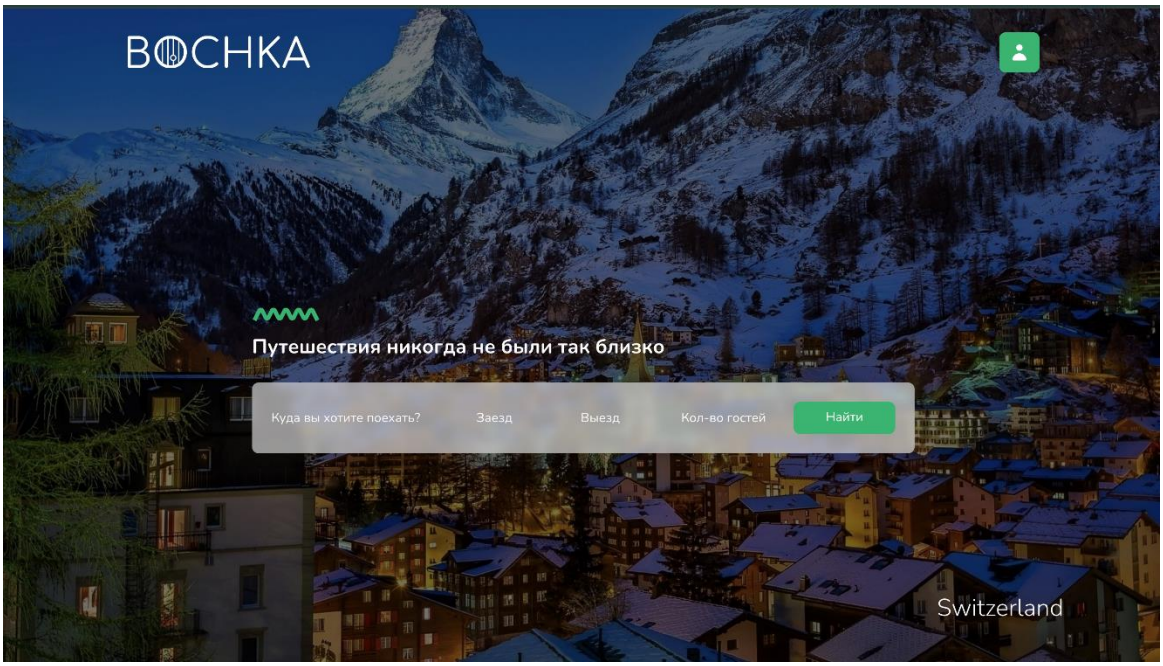

Рисунок 16 - Главная страница

На этой странице пользователь может выбрать нужный ему город, даты заезда, выезда, количество мест и найти отель, подходящий к данному запросу. Помимо этого, пользователь может зарегистрироваться или авторизоваться, наведясь на кнопку пользователя в правом вехнем углу сайта и выбрав нужный вариант (Рисунок 17).

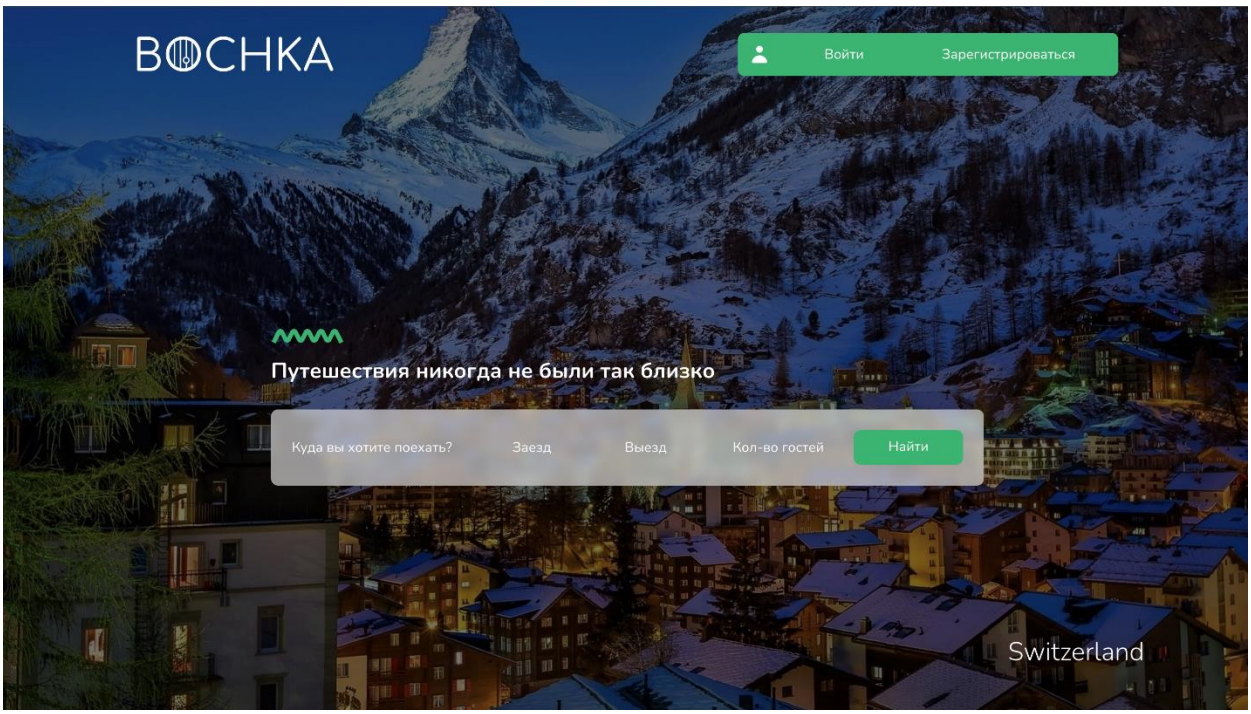

Рисунок 17 - Кнопки авторизации и регистрации

При нажатии на кнопку регистрации, нас перенаправляет на страницу, где пользователь имеет возможность создать новый аккаунт (Рисунок 18).

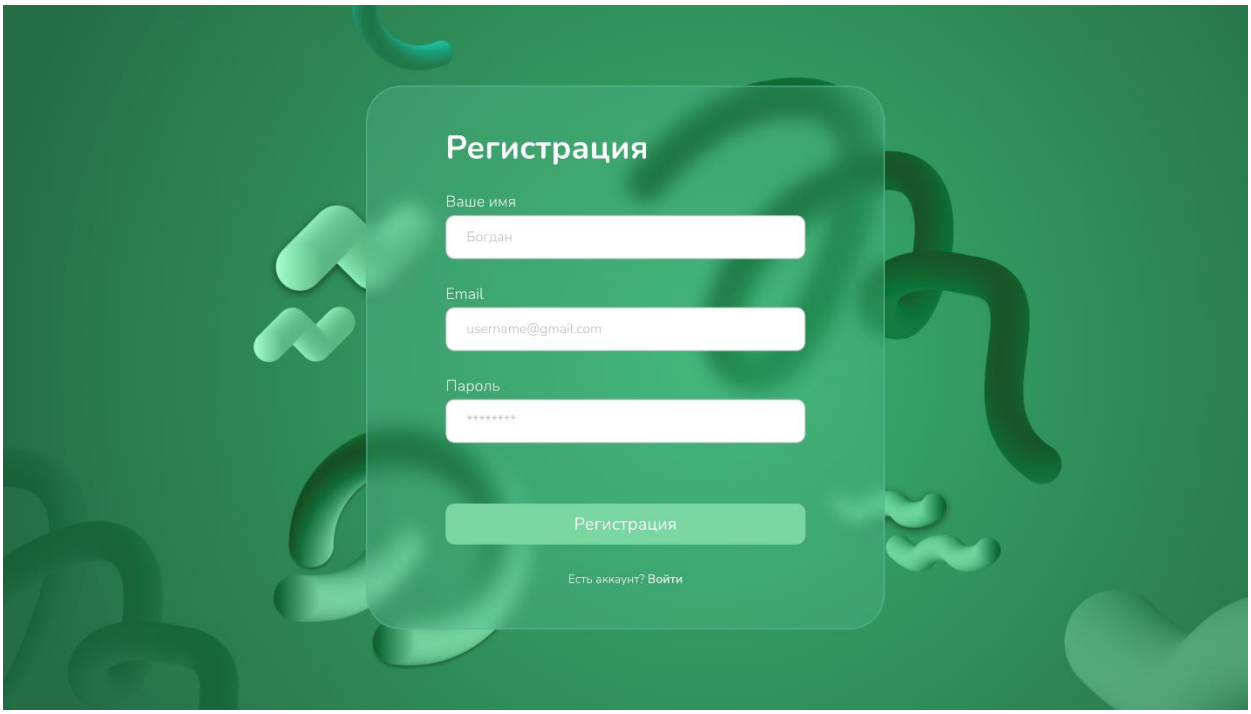

Рисунок 18 - Страница регистрации

Если аккаунт уже имеется, то можно перейти к странице авторизации как с главной страницы, так и со страницы регистрации, нажав на кнопку «Войти» (Рисунок 19).

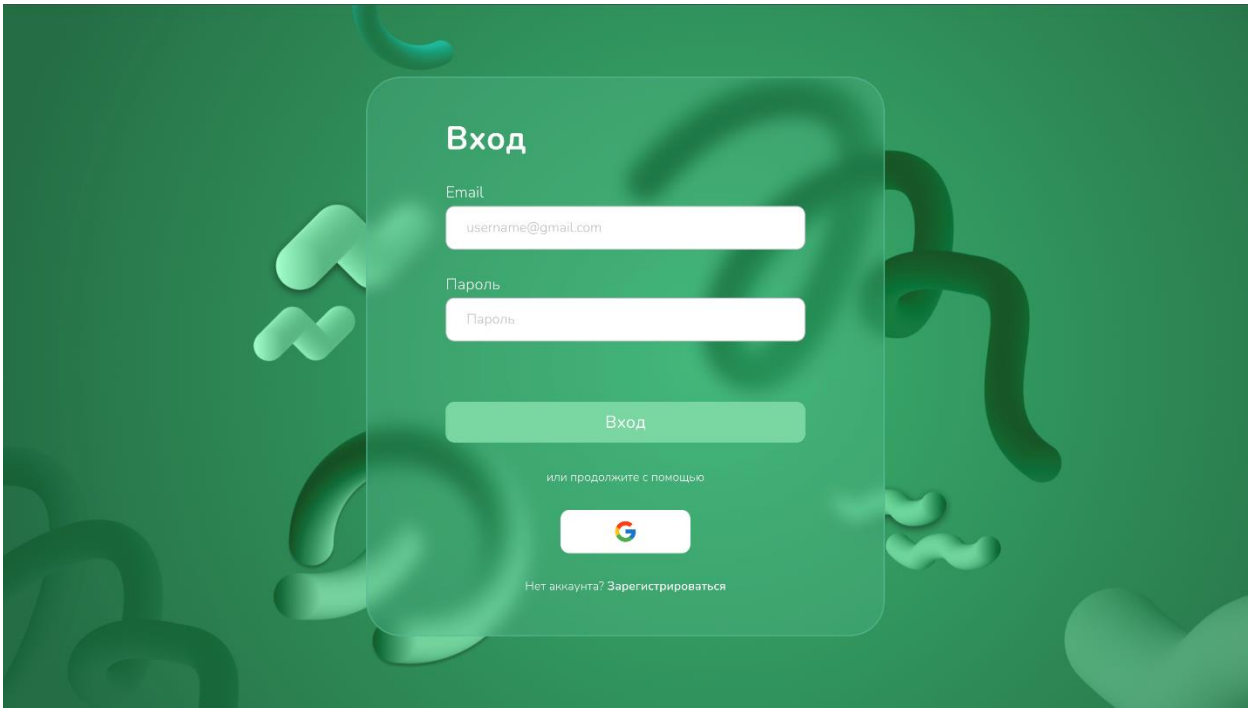

Рисунок 19 - Страница авторизации

Как только пользователь авторизуется, его перенаправит обратно на главную страницу к строке поиска. После выбора нужного города и нажатии на кнопку «Найти» пользователь попадает на страницу со списком различных отелей, находящихся в выбранном городе (Рисунок 20).

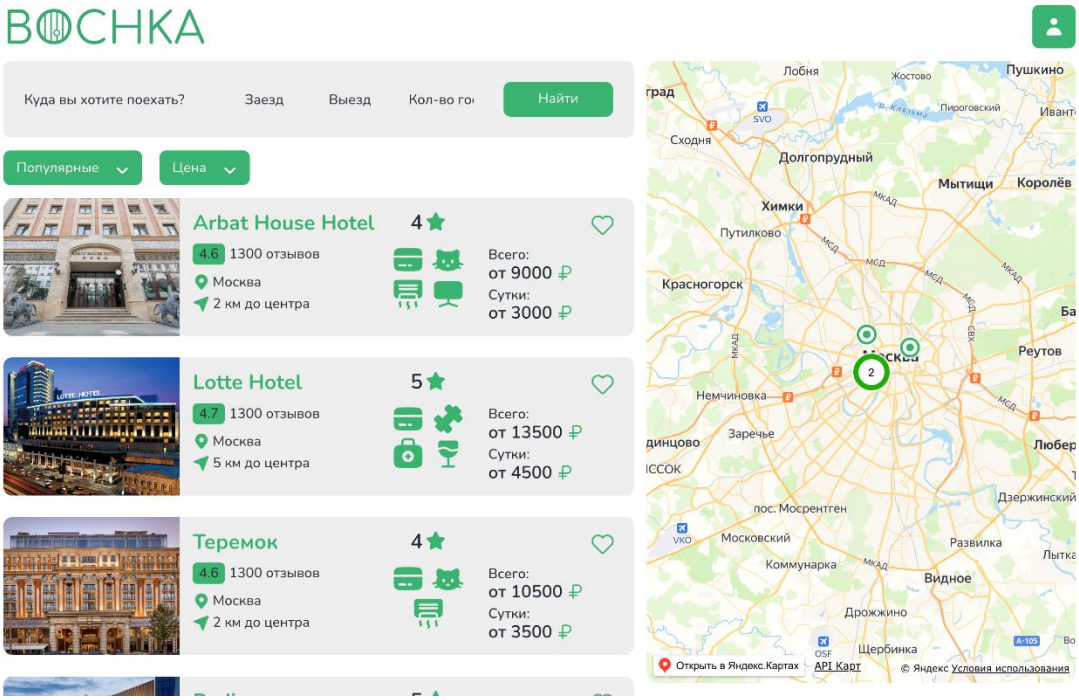

Рисунок 20 - Страница со списком отелей

На данной странице можно отфильтровать полученные результаты, изменяя цену или меняя параметр сортировки результатов. Также на страницу интегрированы Яндекс Карты, на которых отображаются иконки отелей из предоставленного списка.

При нажатии на заинтересовавший нас отель, нас перенаправляет на страницу с подробной информацией о нём и свободных номерах (Рисунки 21-23).

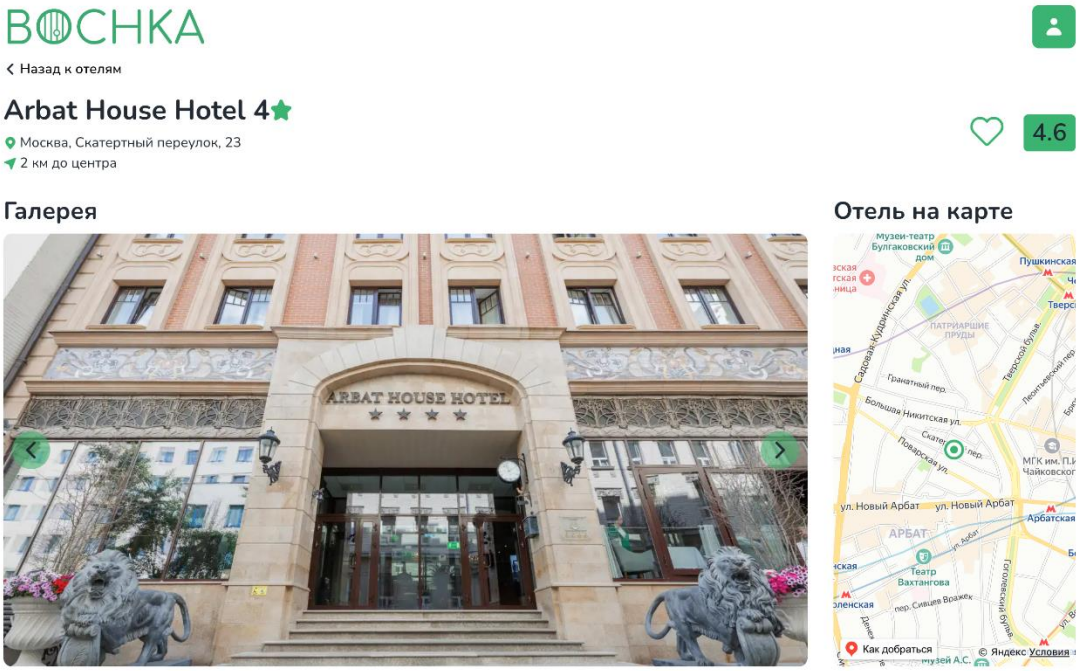

Рисунок 21 - Страница отеля с подробной информацией

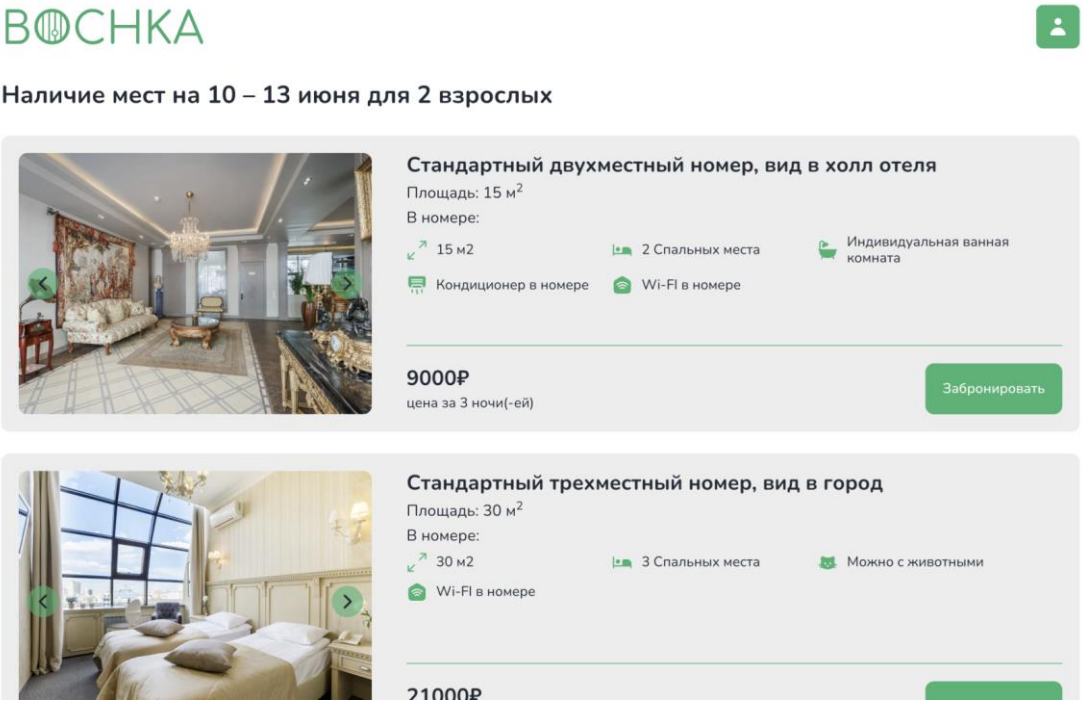

Рисунок 22 - Страница отеля с подробной информацией

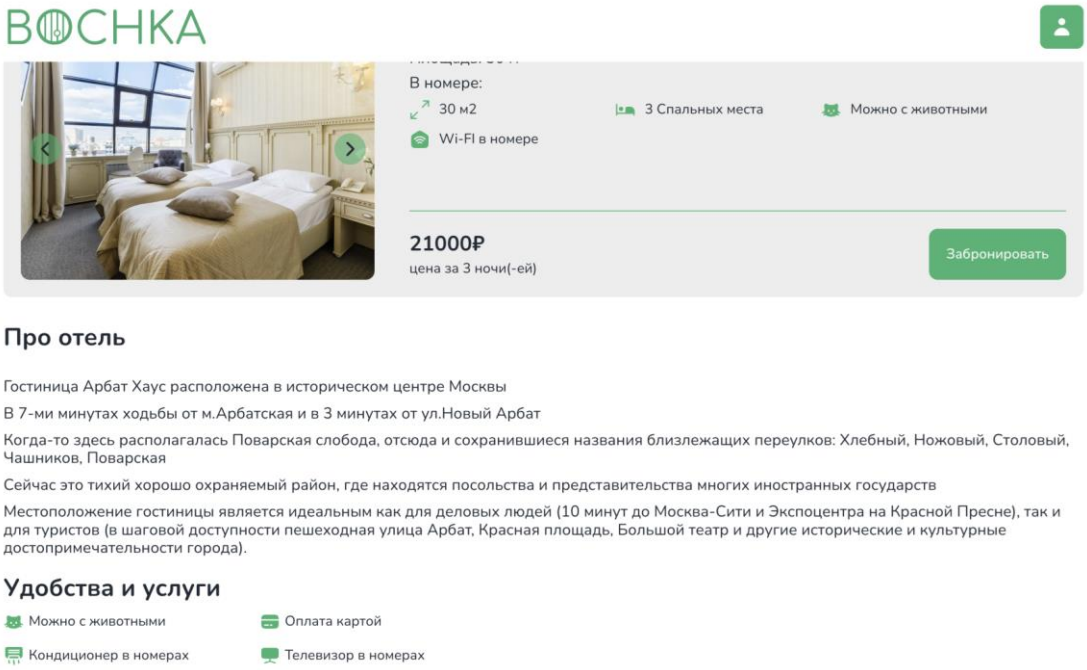

Рисунок 23 - Страница отеля с подробной информацией

Выбрав нужный номер, мы можем нажать на кнопку «Забронировать», после чего откроется страница с полем для ввода информации о госте, на которого оформляется бронь (Рисунок 24).

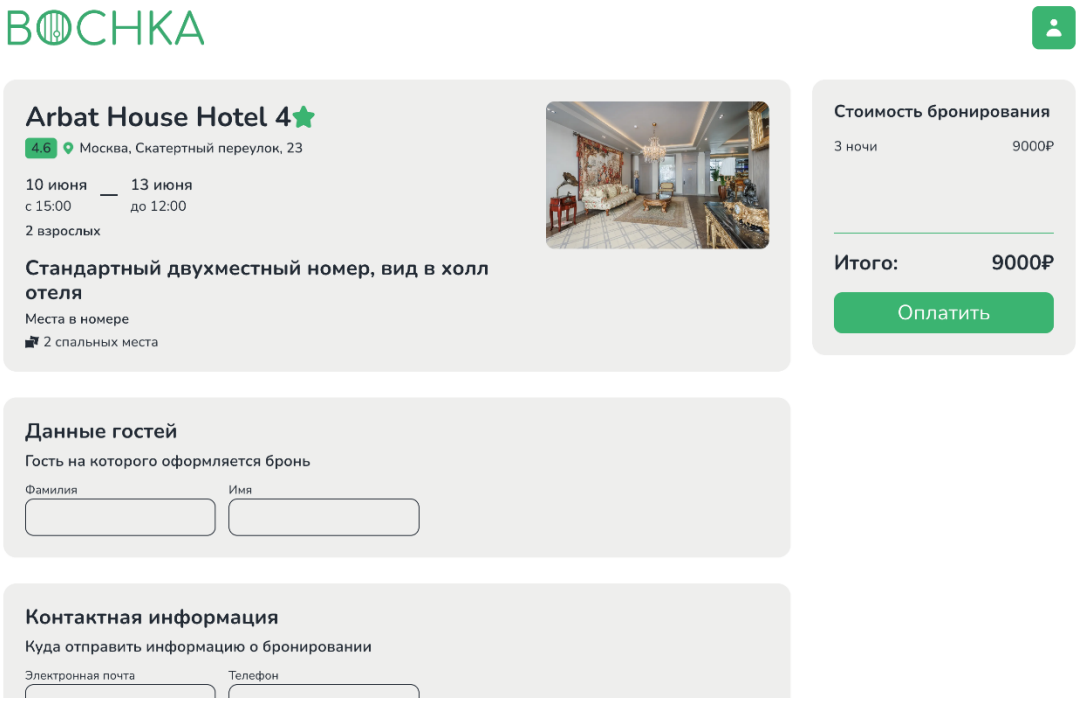

Рисунок 24 - Страница бронирования отеля

После указания данных, мы нажимаем «Оплатить», производится оплата бронирования, и если всё успешно, то на экране появляется окошко об успешно выполненом бронировании (Рисунок 25).

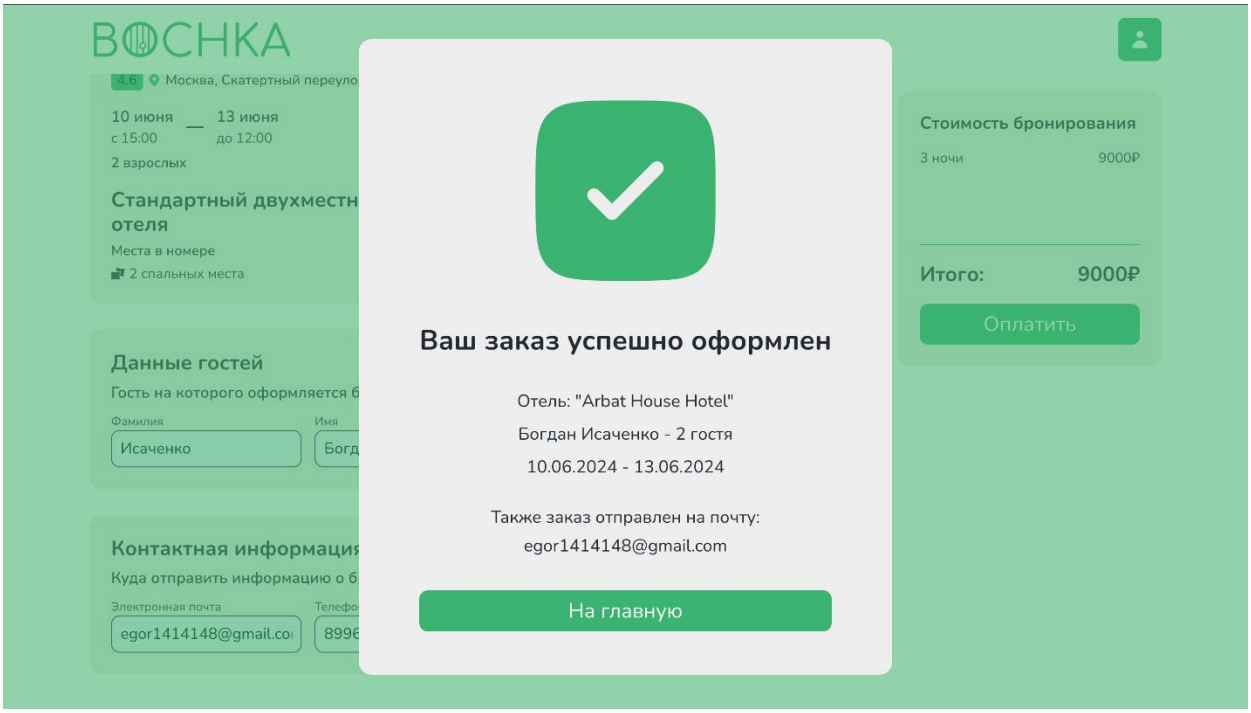

Рисунок 25 - Уведомление об успешно выполненном заказе

Заключение

В ходе данной курсовой работы были выполнены все поставленные задачи. Реализован функционал бронирования отелей, который включает поиск и фильтрацию предложений, просмотр детальной информации о номерах, а также оформление и управление бронированиями.

В работе использовались современные технологии, включая язык программирования Java, фреймворк Spring Boot для серверной части, JavaScript и библиотека React для клиентской части, а также система управления базами данных PostgreSQL. Использование Docker обеспечило изоляцию и переносимость приложений, а Nginx был использован для проксирования запросов и обеспечения поддержки SSL. Для сбора аналитики и мониторинга работы приложения был применен сервис AppMetrica.

В результате выполненных действий была достигнута цель данного курсового проекта – создание функционального прототипа сайта бронирования отелей. Этот прототип предоставляет пользователям возможность искать и бронировать отели онлайн, что делает процесс планирования путешествий более удобным и эффективным. В дальнейшем, на основе разработанного прототипа, можно продолжить развитие и расширение функциональности, учитывая отзывы пользователей и изменяющиеся требования рынка.

Список использованных источников

1. Документация React [Электронный ресурс]. – Режим доступа: URL: https://ru.react.js.org/ – Заглавие с экрана. – (Дата обращения 7.04.2024).

2. Документация Spring Boot [Электронный ресурс]. – Режим доступа: URL: https://docs.spring.io/spring-boot/index.html – Заглавие с экрана. – (Дата обращения 15.05.2024).

3. Документация Swagger [Электронный ресурс]. – Режим доступа: URL: https://swagger.io/docs/ - (Дата обращения 20.05.2024).

4. Вигерс К. Разработка требований к программному обеспечению/ Карл Вигерс, Джой Бити. — М.: Изд-во Русская редакция, 2014. — 736 с.